Comparison of Contemporary Protocols for High-speed Data Transport via 10 Gbps WAN Connections

Dmitry Kachan, Eduard Siemens
Anhalt University of Applied Sciences - Faculty of Electrical, Mechanical and Industrial Engineering,
Bernburger Str. 57, 06366 Köthen, Germany
E-mail: {d.kachan, e.siemens}@emw.hs-anhalt.de

***Abstract*—This work is dedicated to comparison of open source as well as proprietary transport protocols for high-speed data transmission via *IP* networks. The contemporary common *TCP* needs significant improvement since it was developed as general-purpose transport protocol and firstly introduced four decades ago. In nowadays networks, *TCP* fits not all communication needs that society has. Caused of it another transport protocols have been developed and successfully used for e.g. *Big Data* movement. In scope of this research the following protocols have been investigated for its efficiency on 10Gbps links: *UDT*, *RBUDP*, *MTP* and *RWTP*. The protocols were tested under different impairments such as Round Trip Time up to 400 ms and packet losses up to 2%. Investigated parameters are the data rate under different conditions of the network, the CPU load by sender and receiver during the experiments, size of feedback data, CPU usage per Gbps and the amount of feedback data per GiByte of effectively transmitted data. The best performance and fair resources consumption was observed by *RWTP*. From the open source projects, the best behavior is showed by *RBUDP*.**

***Keywords:* high-speed data transport, transport protocol, WAN acceleration, big data.**

I. Introduction

High-speed content delivery is a service that will be more and more demanded by society over the time. Especially it corresponds to reliable delivery, because the movement of Big Data is already part of mission critical services.

High-speed data transmission or more precisely, data transmission with maximum link utilization essentially depends on two factors:

1. link quality: presence of packet losses, abnormal delays or delay jitter;
2. transport protocol: the logic that handles all impairments that interfere data transmission.

Sometimes it is not possible to improve the link quality, moreover, some of impairments that affect data transmission are fundamentally unavoidable e.g. packet delay (OWD or RTT).

The second dependence has another situation: transport protocol is software that can be used on end device and by varying of the algorithms or parameters of the protocol user can achieve significant improvement of transmission rate. The applications that use transport protocols for high-speed data transmission are already available for use on the common PCs. The proprietary algorithms are packed in commercial applications and they are not available for study, investigation and improvement. In contrary, open source approaches give such opportunity, however in most of the cases a commercial application provides better transmission acceleration. There are two reasons of it: proprietary solutions use more effective algorithms or they have better implementation.

In the paper [1] we have compared contemporary proprietary solutions for high-speed data transmission as opaque applications for data transport. As a logical continuation of the work we have presented there, a comparison of the bare protocol performance is presented in this research. Of interest is assertion of pure protocol performance unaffected by accessing to storage, handling data in the application and the processing of that data.

The solutions RBUDP and UDTv4 are protocols that available for free download and currently are provide the fastest transport among open source protocols. RWTP is a proprietary protocol for high speed data transport that has been provided as software library by the Tixel [2] company for test proposals. MTP – is a protocol of the company Data Expedition [3] and was available for trial downloading from the web site of the company. Other providers of high speed data transport application are not provided access to their protocol.

II. Related work

The research in [1] shows the performance of applications based on the investigated protocols. The idea was to transfer 30GiBytes via WAN network in presence of different kinds of impairments and compare the achieved data rates and times of transportation for all solutions. Impressive result within single transport socket has been achieved by transport TIXStream [2]: in presence of 1% of packet loss and 200ms RTT the solution has almost 40% of 10 Gbps link utilization. The solution is based on Reliable WAN Transfer Protocol (RWTP) [3].

Another solution, where an access to its transport protocol was obtained – is ExpeDat, however, with this solution, under the same impairments only 4% of link utilization – 400Mbps were achieved. In this research these two proprietary protocols are compared with open source protocols UDT and RBUDP.

In research [4] Grossman et al. shows that throughput of UDT [5] via 10 Gbps link in presence of 116ms of RTT within a single stream is about 4.5 Gbps. That work also shows that within 8 parallel streams the performance could be increased up to 6.6Gbps. This research shows how real-world data is transported via production network. However of interest is to evaluate whether UDT would be able to handle packet losses in the link as well.

Performance of RBUDP via 10 Gbps connection is presented at the CineGrid 3rd Annual International Workshop [6]. The participants claimed that storage throughput limitation in that particular case was 3.5 Gbps, however the maximal data rate that has been achieved during the workshop did not exceed 1.2 Gbps. The RTT of that link was about 100 ms.

Last example shows implicitly that even if RBUDP was able to achieve better performance – it was limited by 3.5 Gbps, although the link capacity was 10Gbps. However it is not the bottleneck of protocol logics or of protocol implementation or even of the application implementation. It is bottleneck of the hardware. To avoid all issues related to implementation of applications or of the hardware, the evaluation of protocols performance in this work has been performed by transferring of dummy data without accessing to any potentially slow storage.

The packet loss in IP networks is a rather complicated phenomenon that depends on many factors and there is no some universal value that will characterize all the networks in the world. The best way to assess the approximate values of packet losses is through empirical measurements. In [7] Paxson discusses the heterogeneity of packet losses and shows that in 1994, the packet loss ratio between 35 sites in 9 countries was about 2.7 %. These measurements are not really up to date; however the principle knowledge, the author pointed to, is that the distribution of packet losses is not uniform. An assessment of nowadays packet loss dependencies is presented by Wang et al. in [8]. This research describes the tests, that were made across 6 continents between 147 countries, involving about 10 000 different paths. The authors show that across all continents the packet loss rate is less than 2 % for approximately 90 % of continental connections.

For all experiments in the current research, a hardware-based network impairment emulator has been used. In [9] Settlemyer et al. use a hardware emulator to emulate 10 Gbps links, and they compare throughput measurement results of the emulated links with real ones. The maximal RTT of a real link used in the research is 171 ms. The research shows that differences between profiles of both kinds of paths - emulated and real ones – are not significant, and the conclusion is that using emulated infrastructure is a much less expensive way to generate robust and real physical throughput.

III. Testbed Topology Description

The topology for this research replicates the topology in [1] to have a possibility to compare application results with protocol ones, where it is possible.

The scheme of test topology is kept quite simple. In

Fig. 2 the logical connection is presented. The core of the test environment setup is WAN emulator Apposite Netropy 10G [10] that allows an emulation of WAN links with different impairments such as packet loss ratio of up to 100 %, delays of up to 100 000 ms and delay jitter with an accuracy of about 20 ns. By comparison, software-based emulators, such as NetEm, provide an accuracy of about tens of milliseconds and this value is dependent on the hardware and operating system [11]. The Emulator allows a transmission of Ethernet traffic with an overall throughput of up to 21 Gbps on both, copper and fiber optic links.

The experimental setup contains two PC servers, connected via an Extreme Networks Summit x650 10 Gbps Ethernet switch and the WAN Emulator. The topology has been implemented by means of fiber optics with 10 Gbps bandwidth, see

Fig. 2. On the picture, the highlighted connection is the one before network emulator and the connection after it is not highlighted.

There is no background traffic used for experiments, since in the focus of this investigation is the pure protocol performance; the fairness aspects of the protocols are topics for a separate research.

Each server is equipped as follows:

- *CPU: Intel Xeon X5690 @3.47GHz;*
- *RAM: 42 GiBytes (speed 3466 MHz);*
- *OS: Linux CentOS 6.3;*
- *NIC: Chelsio Communications Inc T420-CR, 10Gbps.*

Operating system socket buffers were extended up to:

- */proc/sys/core/net/wmem_max – 64MiBytes*
- */proc/sys/core/net/rmem_max – 64MiBytes*

The MTU size of all network devices along the path has been set to 9000 Bytes.

IV. Experimental Results

The experiment on each data transport solution under consideration consists of 42 consecutive tests. Each test comprises the transfer of dummy data from one server to another via the network emulator with a transmission duration of 180 seconds.

The emulated RTT range is varied from 0 ms up to 400 ms; packet loss ratio reaches up to 2%. One km of optical fiber delays a signal approximately by 5 µs, therefore the RTT 400ms corresponds to 40 000km of fiber optics. The RTT is configured equally across forward and

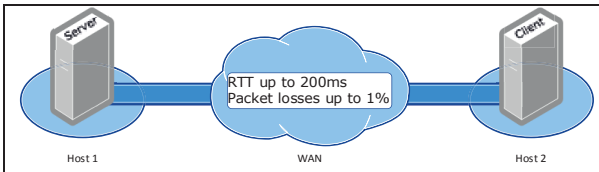

Fig. 1. Logical topology.

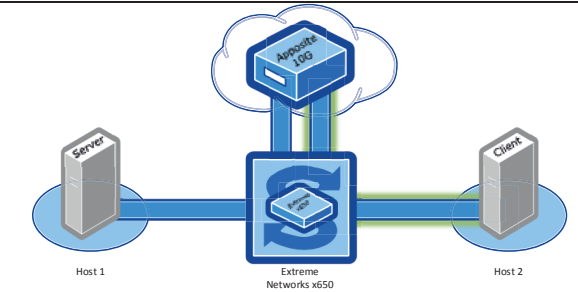

Fig. 2. Technical topology.

backward paths. Thereby 300 ms of RTT will delay signal by 150 ms in one direction and by another 150 in opposite one. The packet losses have a random behaviour with the normal distribution on forward and backward direction. However, it does not correspond to a real packet loss behaviour - in [7] Paxson shows that such behaviour of losses is more complex to handle by protocols than real-world burst packet losses. All the tests were repeated 4 times to avoid inaccuracies, and the best result of each series is presented on the plots.

The result of each test for each set of impairments includes average data rate, average CPU usage by sending host and amount of bytes that have been sent from receiver to sender. From obtained result the following metrics are calculated:

- CPU usage Per Gbps, see (1);

$$U_{per\ Gbps}{}^{i} = \frac{U^{i}}{R^{i}}, \quad (1)$$

where
i – is the number of test;
$U_{per\ Gbps}{}^{i}$ – CPU usage per 1 Gbps;
U^{i} – average CPU usage for i test;
R^{i} – average of data rate for the test.

- MiBytes of feedback per 1GiByte of data, see (2).

$$S_{fpGiB}{}^{i} = \frac{S_{fbk}{}^{i}}{R^{i} \cdot D/8}, \quad (2)$$

where
i – is a number of test;
S_{fpGiB} – size of feedback per GiByte of sent data;
S_{fbk} – size of feedback of whole test;
R^{i} – average of data rate for the test;
D – duration of test.

We consider that test is successful, if the average data rate is higher than 100 Mbps. The metrics have been calculated only for successful tests and average values also have been calculated between successful results.

It was observed that CPU usage behavior of receiver almost in all the cases repeats the usage behavior of sender. The metric CPU usage per Gbps is calculated only for sender. The ratio of average CPU usage on sender side to average CPU usage on reception side is also presented to get a tendency of the CPU consumption differences between the sender and receiver.

CPU consumption has been collected using top application in Linux with period 1s; the size of feedback has been collected using tool dstat with period 1s.

A. *UDTv4*

UDTv4 is a reliable transport protocol based on UDP. It was developed in 2002 at UCI by Yunhong Gu [12]. The protocol is available as a library for Linux with sample applications for file transmission. For the presented tests, the applications have been rewritten in order to avoid data gathering from file system. Only one data stream is used for both data streaming and control communication.

The following settings have been used to achieve maximal performance:

- *MSS – 8800 Bytes*

In Fig. 3 the dependency of data rate from sets of impairments is presented. Decisive factor for the data transmission by *UDTv4* is a value of packet loss. Already by 0.1 % of packet losses the data rate is no higher than 200 Mbps. In case of packet losses more than 0.1 %, the values of data rate are less than 100 Mbps.

In Fig. 4 CPU usage per Gbps by sender is shown. It is worth to mention that *UDTv4* consumes CPU resources significantly even if level of data transmission is quite low. In the best case CPU usage per Gbps is about 28.1 %; in the worst case – 97.8 %. CPU consumption per Gbps at the highest data rate is 28.1 %. The mean value across all experiments is 63.4 % per Gbps.

The CPU consumption at the receiver has the same behaviour; the average of receiver's consumption is 1.1 times less than sender's one.

In Fig. 5 he receiver's feedback statistics is shown. It is easy to observe that the size of feedback not depends on average data rate, however in presences of packet losses its value is decreasing.

The size of feedback per GiByte of data lies in the range from 0.7 MiBytes per GiByte in case of RTT=400 ms and PL=0.1 % and up to 1.5 MiBytes per GiByte in case of RTT=0 ms and PL=0.1 %; in case of best data rate the size of feedback per GiByte is 1.4 MiBytes. The average value across the whole experiment is 1.2 MiBytes per GiByte of data.

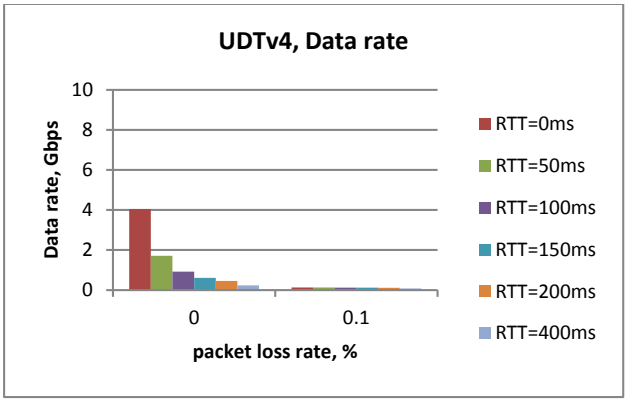

Fig. 3. UDTv4: Data Rate.

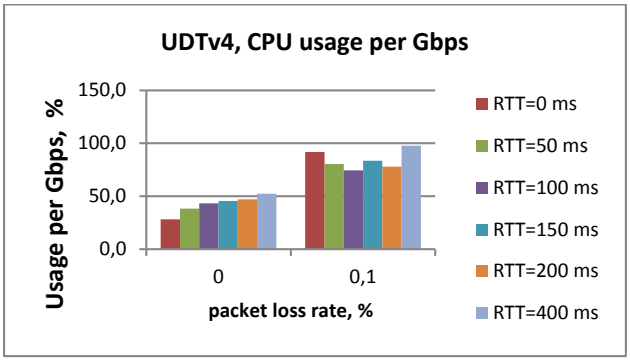

Fig. 4. UDTv4: CPU usage by sender per Gbps.

B. RBUDP

RBUDP [13] is another approach based on *UDP* for reliable data transport. The protocol as well as application was developed also at UCI 2002. In the opinion of the inventors *RBUDP* is a simple replacement of *FTP* for high-speed WAN connections also called Long Fat Pipes (LFP). Both, the application and protocol are open source projects.

The following settings have been used to achieve maximal performance:

- *MSS – 8800 Bytes*

In Fig. 6 he average data rate is shown. The decreasing factor of data rate for RBUDP is RTT. Even without packet losses, with injection of 50 ms of RTT the data rate decreases more than by factor 2. Such a trend is correct also for high values of packet losses e.g. 2 %. However with further increase of RTT the behavior becomes smoother. The protocol is able to transmit data with data rate up to 3 Gbps on connections with RTT about 50 ms. Packet losses also affect the transmission rate, however not that significantly.

In Fig. 7 the distribution of CPU usage per Gbps for each test is shown. The behavior of CPU consumption copies the behavior of average data rate. It is noteworthy that the ratio of CPU consumption on the sender side to CPU consumption by receiver in all cases is approximately equal to 1.2. The values of CPU usage per Gbps lies in the range from 10.4 %, for all the cases with RTT up to 400 ms, and up to 12.7 % by RTT=400 ms and PL=2%. CPU consumption per Gbps by the highest data rate is 10.4%.The average value is about 10.6% per Gbps.

In Fig. 8 the size of feedback per each GiByte of data is shown. The behaviour mostly depends on losses in the network. The values lie in range from 30 KiBytes, in the case without impairments, to 50 KiBytes, in the case with the heaviest impairments. The average value is about 37 KiBytes per GiByte of data.

Each transmission is performed using one UDP and one TCP socket on each side. UDP connection is used for data streaming; TCP – for control communication.

C. MTP

Multipurpose Transaction Protocol (*MTP*) [14] is a proprietary protocol that is used as a core of commercial application *ExpeDat*. The protocol logic is based on *UDP*, and uses a single *UDP* socket on each side of the connection for both data transmission and control information. To get maximal performance from the protocol the following parameters have been used:

- *MSS – 8192 Bytes*
- *Environment variable MTP_NOCHECKSUM=1*

The data rate behavior of *MTP* protocol is shown in Fig. 9. Although the protocol can resist against packet losses, the growing latency decreases the data rate significantly. Even with injection of at least 50 ms of RTT without packet losses the result is 4 times worse than without latency. The interesting behavior of the protocol is presented under condition RTT=0 ms, PL=0.1 % – the data rate exceeds the one without any impairments in the link. Probably the reason for this is increasing of transmission rate by sender when significant delays are detected. In that case decreasing of performance with further increasing of delay is due to buffer limitations.

The CPU usage by sender mostly repeats the behaviour of the data rate. The values of CPU usage per Gbps lie in the range from 8.4 % (RTT= 50 ms; PL= 0.1 %) till 17.6 %

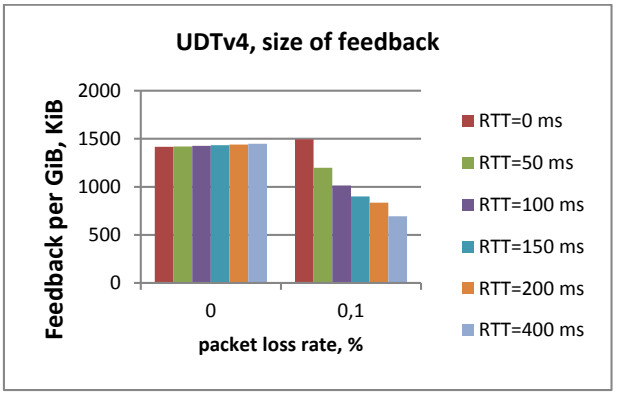

Fig. 5. UDTv4: Size of feedback per GiB of data.

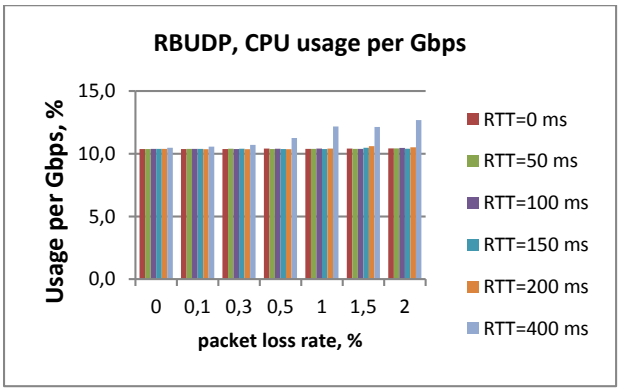

Fig. 7. RBUDP: CPU usage by sender per Gbps.

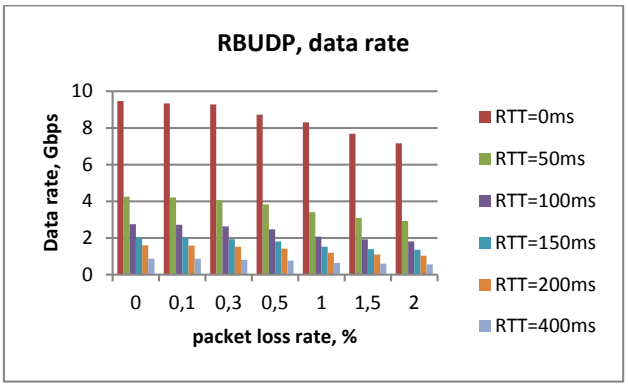

Fig. 6. RBUDP: Data Rate.

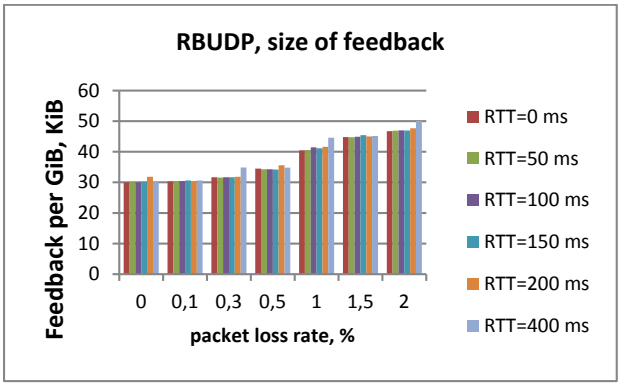

Fig. 8. RBUDP: Size of feedback per GiB of data.

(RTT=50 ms; PL=0 %). The average value is about 11.4 % per Gbps. The matter of interest is CPU consumption of MTP receiver side Fig. 10. MTP is the only one from investigated protocols which CPU consumption on receiver side exceeds the consumption on the sender side. For example by Packet loss 0% the values are approximately in two times bigger. CPU consumption per Gbps by the highest data rate is 9.2%. The average

ratio of sender CPU usage to receiver one is about 0.7.

The feedback size per GiByte of data for MTP is shown in Fig. 11. The biggest size of service information that has been sent back by MTP corresponds to the case when the impairments are absent on the link. The average amount of MiBytes per GiByte of data is about 3.2 MiBytes.

The protocol uses a single UDP connection for both data streaming and control communication.

D. *RWTP*

Reliable WAN Transfer Protocol [15] is a core technology underneath the *TIXStream* data transfer application [16]. The Protocol is based on *UDP* and uses one *UDP* socket on each side for control communication and data transmission. The following parameters have been tuned to achieve maximal performance by *RWTP*:

- *MSS = 8800 Bytes*
- *Receiver buffer size (on both sides) = 1073741824 Bytes (1GiByte)*
- *Sender buffer size (on both sides) = 1073741824 Bytes (1GiByte)*

In Fig. 12 the data rate of *RWTP* is shown. The protocol shows smooth degradation of data rate as impairments grow. The data rate across all the tests have not decreased lower than 1.5 Gbps. Both factors of latency and packet losses are affecting the transmission, however only when they are significant: beginning from 100 ms of RTT and 1 % of packet losses.

The CPU consumption per Gbps by sender is shown in Fig. 13. The behaviour of receiver's CPU consumption repeats the sender's one. The ratio of CPU usage by sender to CPU consumption by receiver is about 1.2.

The values of CPU usage per Gbps lie in the range from 12.4 % (RTT=50ms, PL=0.1%) up to 29.9 % (RTT=400ms, PL=2%). CPU consumption per Gbps by the highest data rate is 12.6 %. The average value is about 15.4 % per Gbps.

The amount of bytes sent back to sender per GiByte of data is shown in Fig. 14. The amount of feedback data grows with growing of both packet losses and RTT. The values of feedback per GiByte are in the range from 5.6 KiBytes (RTT=0ms, PL=0%) up to 6 215 KiBytes (RTT=400 ms, PL=2 %). The average value is about 1 MiByte per GiByte of data.

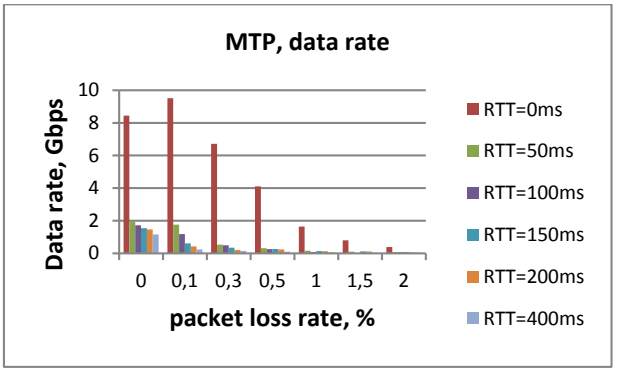

Fig. 9. MTP: Data Rate.

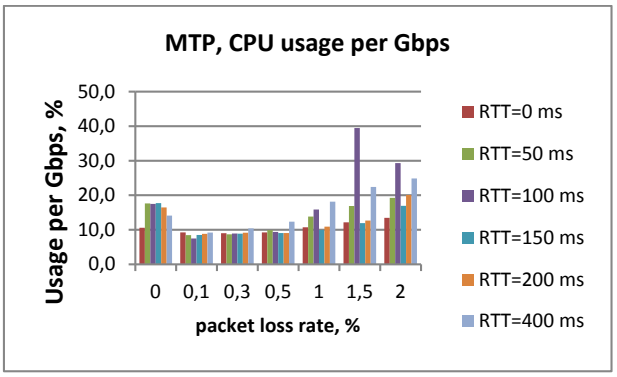

Fig. 10. MTP; CPU usage by receiver per Gbps.

V. Comparison of the Protocols

Only two protocols: *RBUDP* and *RWTP* have passed all the tests successfully – the data rate behaviour has not decreased less than 100 Mbps within the investigated impairments space.

The machines used in experiments are quite powerful, see section 0, to prevent issues with flow control due to a slow receiver.

Analysing the data rates of protocols (Fig. 3 Fig. 6, Fig. 9 and Fig. 12) by the packet loss rate 0 %, the decrease of data rate with the grows of RTT is most likely due to buffering issues. Since without latency all solutions show impressive results, it means that the protocol logic is able to handle the data on such rate appropriately. In *UDTv4* the authors claim that the protocol resizes the buffer automatically based on the value of Bandwidth Delay Product (BDP) [17] while in *RWTP* the buffer have been tuned manually to 1GiByte (the maximal possible value). For *RBUDP* and *MTP* only the system *UDP* buffer adjustment is recommended. The data rate is not decreased significantly on high impairments only by *RWTP* protocol.

The strong resistance against packet losses show *RBUDP* protocol and *RWTP*. *MTP* shows reasonable data rate in absence of latencies, however only till 0.5% of packet losses. Packet losses affect behaviour of *UDTv4* dramatically - with injection of at least 0.1 % of packet losses the protocol decreases its data rate down to 130 Mbps which is 1.3 % of its possible link capacity.

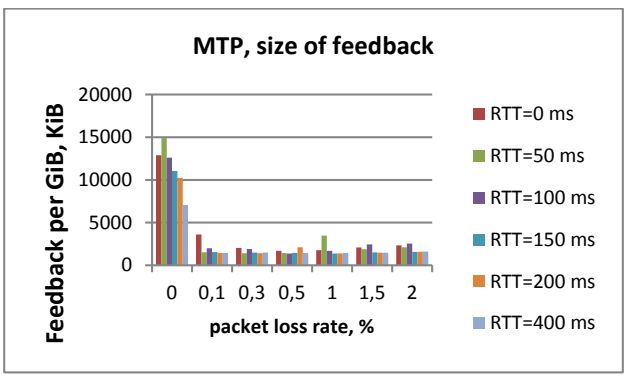

Fig. 11. MTP: Size of feedback per GiB of data.

It is possible to compare the behaviour of the pure protocol with behaviour of the solution based on that protocol. The performance of application based on *RWTP* and on *MTP* are presented in [1]. In case of *TIXStream*/*RWTP*, the experiment with *RWTP* showed better performance than MTP. The difference is most likely caused by application overhead due to the storage access and another system resources that are not responsible for data transmission. In case of *ExpeDat*/*MTP* the result is better by the application. Possible reason of it is not using of flag "-N 25" that shows that there are significant losses on the link by protocol performance test. However, this flag would not affect the case without packet losses, although even without the losses, the data rate by the application is higher than in the pure protocol case.

CPU consumption of every protocol except *MTP* is higher on the sender side than on the receiver side. The overhead of CPU usage for all solutions is not that significant – about 1.2 times. The only *RWTP* and *UDTv4* showed CPU usage higher than 100% on the sender side – it means that the protocol use more than one core for data transmission. This fact allows to avoid CPU bottlenecks for high-speed data transmission. In case of *RBUDP,* it is observable that under set of impairments where RTT=0 ms; PL=0 % or PL=0.1 % (see Fig. 6) the average CPU consumption is close to 100 % but does not exceed that value. Probably the data rate in these cases is not higher because of CPU bottleneck. This observation corresponds also for *MTP* behaviour.

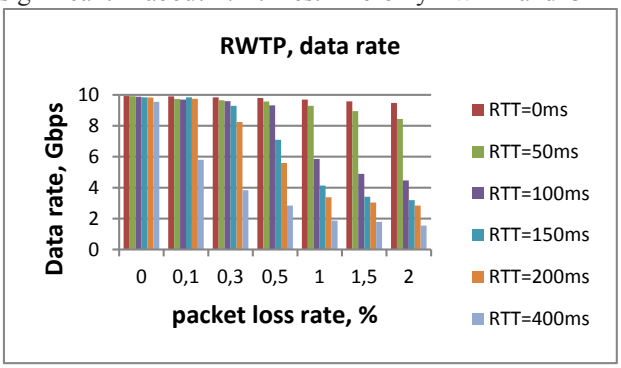

Fig. 12. RWTP: Data Rate.

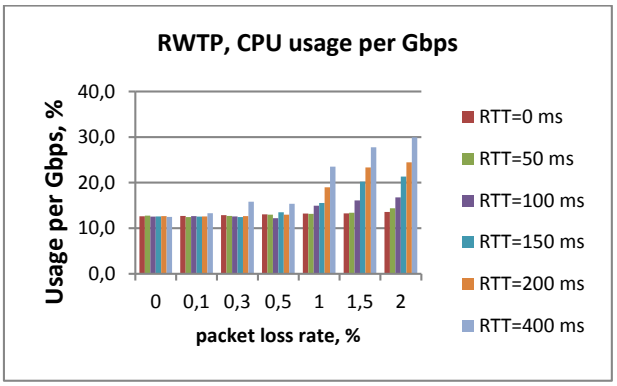

Fig. 13. RWTP: CPU usage by sender per Gbps.

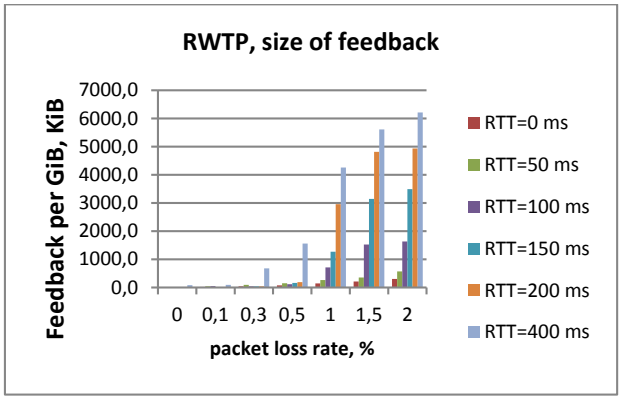

Fig. 14. RWTP: Size of feedback per GiB of data.

The lowest average CPU consumption per Gbps is shown by *RBUDP*. However at the maximal data rate, the lowest consumption is achieved by *MTP* – 9.2 %. The highest – by *UDTv4:* 28.1 %. In all protocols, except *MTP*, with growing impairments the relative CPU consumption is increasing. It is most likely due to necessity in heavy retransmissions. *MTP* is the only protocol that has decreasing trend of CPU consumption on raising transmission rate, however the data rate of *MTP* in presence of impairments is quite low.

The lowest values of feedback data rate have been obtained within the experiment with *RBUDP* – across all experiments the values are about tens of KiBytes per GiByte of transmitted data. The reason of it is that, the retransmissions are done in "blast", at the end of entire data transmission. The biggest value has been obtained by *MTP*. Probably this explains poor behaviour of the protocol. It is considered that packet losses affect feedback channel significantly less than traffic in forward direction [7]. In our experiments the packet losses have somewhat unrealistic random behavior with the normal distribution across both forward and backward link, what corresponds to extreme network conditions. The fair behavior in feedback has RWTP – without losses the feedback is measured in KiBytes and with growing impairments the value increases up to few MiBytes per GiByte. *UDTv4* has less feedback in presence of packet losses than without them. Within entire experiment under *UDTv4* the size of feedback is always in the range of few MiBytes.

Besides performance characteristics, usability of the transports are also of interest. The main disadvantage among all tested solutions is that some parameters should be predefined to reach higher performance. So the MSS size adjustment makes significant improvement of performance for all transports. The disadvantage of *RWTP* and *RBUDP* is that the user should also pre-define desired send rate and if the data rate will be higher than available throughput – the congestion occurs from which the software doesn't recover. So the user should use some third party software like *LTest* [18] to investigate the link before data transmission. In case of *RWTP,* user should manually define the buffer size, if the default one is not big enough.

The only one protocol that uses more than one socket per session on each side is RBUDP. It is disadvantage since the protocol is not all-sufficient and needs support of one more reliable transport protocol - TCP.

VI. Conclusion

Reasonable performance has been achieved by *RBUDP* and *RWTP* protocols. *RBUDP* acts sometimes better than some proprietary solutions for high-speed data transport in [1]. *RWTP* shows the best results within this research: it

has resistance against packet losses and latency and it consumes reasonable amount of resources, however, the buffers should be predefined. It consumes more than one core which helps to achieve better performance than other solutions. The disadvantage of *RWTP* is that it is proprietary protocol. The *RBUDP* shows worse performance than RWTP, however, it is open source, which makes it more attractive to use it in some projects, or even to improve the algorithms of the protocol. The specific of RBUDP – requesting of all lost packets after receiving of all the data – allows minimizing the size of feedback, however such specific make impossible use of the protocol for streaming applications.

The size of feedback is not the important parameter as long as it comprises permilles of the net forward data amount, however if it is tens of MiBytes in case of transmission without impairments, as it showed by MTP in Fig. 11, the efficiency of that feedback is questionable.

The investigation of CPU usage showed that most likely the transports RBUDP and MTP would provide higher performance if they would use more than one core for sending of a data, or even if the sending of a data and receiving of a feedback by sender will be distributed between two separate cores.

VII. Future work

The logical continuation of that research is a comparison of the protocols overhead in the forward direction. It is of interest to assess the relation of a throughput on the application layer – pure user data – to throughput on the network layer – user data including service information.

References

[1] D. Kachan, E. Siemens, V. Shuvalov, "Comparison of Contemporary Solutions for High Speed Data Transport on WAN 10 Gbit/s Connections," Journal of Communication and Computer, vol. 10, Nr. 6, pp. 783-795, 06 2013.

[2] Tixel GmbH, Tixel, [Online]. Available: http://www.tixeltec.com/ps_tixstream_en.html. [Accessed 20 10 2012].

[3] S. Höhlig, "Optimierter Dateitranfer über 100 Gigabit/s," In Proceeding of the 100 Gigabit/s Workshop in Mannheim, September 2011.

[4] R. L. Grossman, Y. Gu, X. Hong, A. Antony, J. Blom, F. Dijkstra, C. de Laat, "Teraflows over Gigabit WANs with UDT," Journal of Future Computer Systems, pp. 501-513, 2005.

[5] R. L. G. Y. Gu, "UDP-based datatransfer for high-speed wide area networks," Computer Networks, vol. 51, no. issue 7, pp. 1465-1480, May 2007.

[6] L. Herr, M. Kresek, "Building a New User Community for Very High Quality Media Applications On Very High Speed Networks," GLIF 2008, 1 10 2008. [Online]. Available: http://czechlight.cesnet.cz/documents/publications/network-architecture/2008/krsek-cinegrid.pdf. [Accessed 05 02 2013].

[7] V. Paxson, "End-to-End Internet Packet Dynamics," Networking, IEEE/ACM Transactions, vol. 7, no. Issue 3, pp. 277-292, 1999.

[8] Y. A. Wang, C. Huang, J. Li, K. W. Ross, "Queen: Estimating Packet Loss Rate between Arbitrary Internet Hosts," Proceedings of the 10th International Conference on Passive and Active Network Measurement, pp. 57-66, 2009.

[9] B. W. Settlemyer, N. S. V. Rao, S. W. Poole, S. W. Hodson, S. E. Hick, P. M. Newman, "Experimental analysis of 10Gbps transfers over physical and emulated dedicated connections," proceeding of Computing, Networking and Communications (ICNC), pp. 845-850, 2012.

[10] Apposite Technologies, "Apposite," [Online]. Available: http://www.apposite-tech.com/index.html. [Accessed 20 10 2012].

[11] A. Jurgelionis, J.-P. Laulajainen, M. i. Hirvonen, and A. I. Wang, "An Empirical Study of NetEm Network Emulation Functionalities," 20. ICCCN, no. ISBN 978-1-4577-0637-0, pp. 1-6, 2011.

[12] R. G. Y. Gu, "Udt: a high performance data transport protocol," Doctoral Dissertation, University of Illinois, Chicago, 2005.

[13] E. He, J. Leigh, O. Yu, and T. A. DeFanti, "Reliable Blast UDP : Predictable High Performance Bulk Data Transfer," Proceedings of IEEE Cluster Computing, pp. 317-324, Sept. 2002.

[14] Data Expedition, Inc, "Data Expedition," [Online]. Available: http://www.dataexpedition.com/downloads/DEI-WP.pdf. [Accessed 25 10 2012].

[15] Tixel GmbH, "White Papers and Reports," [Online]. Available: http://www.tixeltec.com/papers_en.html. [Accessed 15 11 2012].

[16] Tixel GmbH., "Tixel news," [Online]. Available: http://www.tixeltec.com/news_en.html. [Accessed 20 10 2012].

[17] R. L. G. Y. Gu, "UDTv4: Improvements in Performance and Usability," in Networks for Grid Applications, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, 2 ed., Berlin Heidelberg, Springer, 2009, p. 9.

[18] E. Siemens, S. Piger, C. Grimm, M. Fromme., "LTest – A Tool for Distributed Network Performance Measurement," Proc. Consumer Communications and Networking Conference, 2004. First IEEE, pp. 239-244, 2004.

[19] E. S. D. Kachan, "AvBandTest - a Testing Tool for Implementations of Available Band width Estimation Algorithms," in Proceedings of 1st International Conference on Applied Innovations in IT, Kothen, 2013.